REMARKS

Claims 1-6, 8-13, 15-20, and 22-29 are pending. Please amend claims 1 and 6. No claims are canceled, added, or withdrawn.

As a preliminary matter, Applicant appreciated the opportunity to discuss the September 13, 2005 Action with the Examiner during a December 05, 2005 telephone conversation. During that conversation, the Examiner assured Applicant that the Action would be treated as a non-final Action, as indicated in the Office Action Summary and independent of indication at the end of the Action (page 12, section 30) that the action was final. Additionally, Applicant was pleased to determine that the Examiner was interested in discussing the claimed subject matter in accord with Applicant's desire to move this application towards allowance.

In view of the following remarks, withdrawal of the rejections to the pending claims is respectfully requested.

Claim Amendments

Claim 1 was amended to more particularly point out that the operations "for providing thread scheduling in a device", as claim 1 recites, are implemented by a computer. To this end, the preamble of claim 1 was changed from "[a] method for providing thread scheduling in a device" to "[a] computer-implemented method for providing thread scheduling in a device". Claim 6 was amended to correct a grammatical error.

35 USC §103(a) Rejections

Claims 1-6, 8-13, 15-20, and 22-29 stand rejected under 35 USC §103(a) as being unpatentable over U.S. Patent No. 6,308,279 to Toll et al

("Toll") in view of US patent application number 6,584,571 to Fung. Applicant respectfully traverses these rejections.

Claim 1 recites in part "responsive to a determination that there are no threads to execute, deactivating one or more of the hardware elements and the program modules for a dynamic variable amount of time, the dynamic variable amount of time being independent of the predetermined periodic rate and being based on a sleep state of a set of threads in a sleep queue." In addressing these claimed features, the Action asserts that they are taught by Toll (i.e., at column 2, lines 32-34, and column 3, lines 8-30) and by Fung (i.e., column 6, lines 14-30 and 45-51). Applicant respectfully disagrees for the following reasons.

The Action at page 10, section 24, agrees that Toll teaches a thread scheduling mechanism that schedules threads based on a predetermined periodic rate such as a system tick. The Action at page 11, section 25, also agrees that Toll does not explicitly disclose the claimed feature of "deactivating one or more of the hardware elements and the program modules for a dynamic variable amount of time, the dynamic variable amount of time being independent of the predetermined periodic rate and being based on a sleep state of a set of threads in a sleep queue", as claim 1 recites. However, the Action indicates that the above recited claimed feature is suggested by Toll since Toll describes that already deactivated hardware elements can be reactivated responsive to receipt of a break event (i.e., a user opening a laptop lid, a keypress, etc.) that could occur at any time (i.e., the user initiated break event is not keyed to a system clock, as is

the time slice that is used to schedule threads). Applicant respectfully submit that this conclusion is unsupportable.

With respect to Toll's "break events", Toll teaches at lines 7-18 of col. 3 that deactivated hardware elements can be reactivated responsive to predetermined "break events". Toll is completely silent with respect to any description of what can represent such a "break event". However, in view of the fact that Toll describes that core clocks may be left running to process "snoops", it is likely that a "break event" at least includes "snoop" request processing. More particularly, Toll at col. 3, lines 4-18, describes that when all threads are asleep, core clocks may be left running to process a "snoop", in which case the system of Toll will respond normally. Toll does not describe what a "snoop" request is, or what is involved in processing a "snoop" request. However, Applicant respectfully submits that a "snoop" request is likely associated with a core clock system tick that triggers a thread scheduling mechanism to "snoop" for threads to activate. When a hardware element such as a processor is deactivated, such processing requires reactivation of the deactivated processor to execute thread scheduling to "snoop" the sleep queue and determine if there are any threads ready for execution.

In view of the above, Toll's system to process a "snoop" request means that a deactivated hardware element (i.e., the processor) is woken up (reactivated) responsive to a system tick (a "snoop" event) that is generated at a predetermined periodic rate. Thus, these particular snoop request processing teachings of Toll do not teach or suggest "deactivating one or more of the hardware elements and the program modules for a dynamic

variable amount of time [that is] independent of the predetermined periodic rate", as claim 1 recites.

Additionally, Toll's period of time that is represented by the core clock system tick that causes a "snoop" request is clearly not "based on a sleep state of a set of threads in a sleep queue" (as claim 1 recites), but rather based on a periodic hardware timer interrupt (i.e., a system tick). Thus, Applicant respectfully submits that that even after hardware elements are deactivated, the system of Toll implements a conventional system thread scheduling mechanism to process "snoop" requests. Such conventional system thread scheduling mechanisms are clearly described in the Background section of Applicant's specification.

Moreover, besides a "break event" including a "snoop" request (as discussed above), it is also possible that a "break event" may represent other conventional types of break events such as user key-presses, etc. (conventional processing of which is also clearly described in the Background section of Applicant specification). In view of this, and for purposes of discussion, assume that the system of Toll can be configured so that it will no longer process "snoop" requests (snoop request processing is a configuration clearly supported by the teachings of Toll). In this alternate configuration, assume arguendo that Toll **deactivates** all timers (including snoop request generating core clock timers) **for reactivation responsive to an external event such as a user key-press that occurs only after some amount of time elapses since the hardware elements were deactivated**. In this particular configuration, Applicant agrees that the amount of time that would elapse between hardware element deactivation and reactivation

responsive to a user initiated an event would be independent of the predetermined amount of time at which Toll schedules threads. However, it is also very plain that this amount of time (between Toll's system deactivation of hardware elements and reactivation responsive to a user action) is also completely independent of Toll's in-device thread scheduling operations. The result of such deactivation **essentially removes and stops all thread scheduling activities in the system of Toll - potentially forever** (unless a user intervenes).

In the above scenario, threads that are sleeping (not yet ready for execution) in a queue of Toll, **will never be woken up regardless of their respective sleep times** unless a user intervenes. This is because the amount of time between deactivation and reactivation in the system of Toll, while independent of the predetermined period rate, is clearly not "based on a sleep state of a set of threads in a sleep queue", as claim 1 recites, but rather based on user activity that is completely separate from the deactivated thread scheduling mechanism of Toll. Clearly, this does not teach or suggest "providing thread scheduling in a device", as claim 1 recites, but instead teaches removing thread scheduling in a device. In contrast to these teachings of Toll, the features of claim 1 always "provid[e] thread scheduling and a device". In a system according to claim 1, thread scheduling is not disabled and at least one thread sleeping in a sleep queue of the invention will always be reactivated after passing of a "dynamic variable amount of time [that is] based on a sleep state of a set of threads in a sleep queue", as claim 1 recites, always "providing thread scheduling in a device".

For each of the above reasons reasons, Toll does not provide any description that teaches or suggests "providing thread scheduling in a device" that "responsive to a determination that there are no threads to execute [will] deactivating one or more of the hardware elements and the program modules for a dynamic variable amount of time [that is] independent of the predetermined periodic rate [the rate at which threads are scheduled] and being based on a sleep state of a set of threads in a sleep queue", as claim 1 recites.

Modifying Toll with Fung's system that returns to an active power mode in response to an increased level of system activity (e.g., threads with sleep times that have expired) does not cure the above described deficiencies of Toll. Applicant respectfully submits that Fung merely describes a system wherein threads are deactivated for non-dynamic amounts of time that are completely dependent on Fung's system tick rate (as are Toll's "snoop" request processing operations). This is similar to the conventional systems described in Applicant's background section wherein a thread's quantum is always a multiple of the time between system ticks.

In view of the above, the system of Toll combined with Fung does not teach or suggest each and every element claim 1, and therefore, not obvious over the cited combination. Accordingly, withdrawal of the 35 USC §103(a) rejection of claim 1 is respectfully requested.

Claims 2 – 7 depend from claim 1 and are allowable over the cited combination solely by virtue of this dependency. Accordingly, and for this reason alone, the 35 USC §103(a) rejections of claims 2-7 over Toll is improper and should be withdrawn.

Additionally, claims 2-7 include additional subject matter that is not taught or suggested by the cited combination.

For example, claim 6 recites:

- "resetting the system timer to generate the notification after the dynamic variable amount of time has elapsed since the deactivating",
- "receiving the notification after the dynamic variable amount of time has elapsed since the deactivating",
- "responsive to the receiving: resetting the system timer to generate the notification at the predetermined periodic rate" and
- "activating the one or more of the hardware elements and the program modules."

In addressing these claimed features, the Action at section 11, page 5, asserts that "resetting the system timer to generate the notification after the dynamic variable amount of time has elapsed since the deactivating" is taught by Toll at column 9, line 6-8. However, there are no such columns and lines in Toll. Instead, the description of Toll stops at column 8. Even in view of this, Applicant respectfully submits that Toll, as a whole, merely teaches: (1) that a core timer may not be turned off and that responsive to a snoop request, deactivated hardware elements may be reactivated to determine if any sleeping threads are ready to be executed; and (2) that the core timer may be turned off and then reactivated at some other time either responsive to a user action. This does not teach or suggest modifying the core timer to generate hardware interrupts at different rate than that initially used to schedule threads.

Specifically, nowhere does Toll provide any description that teaches or suggests that the core timer is first reset to no longer provide hardware interrupts at the predetermined periodic rate (the rate at which threads are scheduled), but to instead provide a hardware notification at a different rate equivalent to "the dynamic variable amount of time". Moreover, nowhere does Toll teach or suggest that after the core timer has been modified to generate hardware interrupts at rates different than an initial thread scheduling rate, that the hardware timer is again reset to generate hardware interrupts at the initial scheduling rate. The Action does not rely on Fung for these teachings. However, Applicant respectfully submits that Fung is completely silent with respect to any teachings or suggestions of these claimed features. Thus, Toll modified with the teachings of Fung fails to teach or suggest all of the claimed features of claim 6.

Accordingly, withdrawal of the 35 USC §103(a) rejection of claim 6 is respectfully requested.

Claim 8 recites:

- "scheduling one or more threads at a predetermined periodic rate",
- "determining whether or not there are any threads to execute",
- "responsive to a determination that there are no threads to execute, deactivating one or more of the hardware elements and the program modules for a dynamic variable amount of time, **the dynamic variable amount of time being based on a sleep state of a set of threads in a sleep queue and independent of the predetermined periodic rate**", and

- "activating the one or more of the hardware elements and the program modules only when the operating system needs to perform an action selected from a group of actions comprising scheduling a thread for execution upon expiration of the dynamic variable amount of time since the deactivating, or upon receipt of an external event that is not a system timer event" (emphasis added).

For the reasons already articulated with respect to claim 1, Toll in view of Fung does not teach or suggest these recited features of claim 8.

Accordingly, withdrawal of the 35 USC §103(a) rejection of claim 8 is respectively requested.

Claims 9-14 depend from claim 8 and are allowable over the cited combination solely by virtue of this dependency.

Accordingly, withdrawal of the 35 USC §103(a) rejection of claims 9-14 is respectively requested.

Moreover, claim 12 includes additional features that are not obvious over Toll in view of Fung. Specifically, claim 12 recites:

- "wherein the scheduling further comprises setting a system timer to the predetermined periodic rate, the predetermined periodic rate corresponding to a thread scheduling accuracy", and
- "wherein the deactivating further comprises resetting the system timer to generate a notification after the dynamic variable amount of time has elapsed since the deactivating."

For the reasons already discussed above with respect to claim 6, Toll in view of Fung does not teach or suggest these claimed futures.

For these additional reasons, withdrawal of the 35 USC §103(a) rejection of claim 12 is respectfully requested.

In another example, claim 13 recites:

- "wherein the deactivating further comprises resetting a system timer to generate a notification after the dynamic variable amount of time has elapsed, the dynamic variable amount of time being a maximum amount of time that a thread can yield to other threads before needing to be scheduled for execution", and
- "wherein the activating further comprises resetting the system timer to the predetermined periodic rate to provide substantial thread scheduling accuracy."

In addressing these claimed features, the Action asserts that "wherein the deactivating further comprises resetting a system timer to generate a notification after the dynamic variable amount of time has elapsed, the dynamic variable amount of time being a maximum amount of time that a thread can yield to other threads before needing to be scheduled for execution" is taught by Toll at column 9, lines-8-18. However, as indicated above with respect claim 6, the cited portions of Toll do not exist as Toll's description stops at column 8. Even in view of this, and for the reasons already discussed above, the cited combination is completely silent with respect to these claimed elements.

Accordingly, withdrawal of the 35 USC §103(a) rejection of claim 13 is respectfully requested.

Claim 15 recites:

- "determining at a periodic rate whether or not there are any threads to execute", and
- "responsive to a determination that there are no threads to execute, deactivating one or more of the program modules and the hardware elements for a dynamic variable amount of time, **the dynamic variable amount of time being independent of the periodic rate, the dynamic variable amount of time being based on a sleep state of a set of threads in a sleep queue**" (emphasis added).

For the reasons already discussed, Toll in view of Fung does not teach or suggest these recited features. Withdrawal of the 35 USC §103(a) rejection of claim 15 is respectfully requested.

Claims 16-21 depend from claim 15 and are allowable over Toll solely by virtue of this dependency. Accordingly, withdrawal of the 35 USC §103(a) rejection of claims 16-21 is respectively requested.

Moreover, claim 19 includes additional features that are not taught or suggested by Toll in view of Fung. Specifically, claim 19 recites:

- "in the deactivating, **configuring a system timer to send a first timer interrupt after the dynamic variable amount of time has elapsed, the dynamic variable amount of time being a maximum amount of time that a first thread can yield to a second thread before the first thread needs to be executed**", and

- "responsive to receiving the first timer interrupt:
    - (a) configuring the system timer to send a second timer interrupt at the periodic rate" and
    - "(b) activating the one or more of the program modules and at the hardware elements to determine if there are any threads to execute" (emphasis added).

At least for the reasons already discussed above, Toll in view of Fung does not teach or suggest these claimed futures. Withdrawal of the 35 USC §103(a) rejection of claim 19 is respectfully requested.

Claim 22 recites:

- "scheduling threads for execution at a periodic time interval",
- "determining that there are no threads to execute", and
- "wherein the HAL, responsive to the determining, comprises computer-executable instructions for deactivating, for a dynamic variable amount of time, one or more of the scheduler, the hardware elements, the one or more operating system program modules, and the application program modules, the dynamic variable amount of time being independent of the periodic time interval and being based on a sleep state of a set of threads in a sleep queue."

For the reasons are discussed above with respect claim 1, Toll in view of Fung do not teach or suggest these recited features. Withdrawal of the 35 USC §103(a) rejection of claim 22 is respectfully requested.

Claims 23-29 depend from claim 22 and are allowable over the cited combination solely by virtue of this dependency. Accordingly, the 35 USC §103(a) rejections of claims 23-29 should be withdrawn.

Additionally, for the additional the reasons already discussed, withdrawal of the 35 USC §103(a) rejections of claim 29 is respectfully requested.

Conclusion

Pending claims 1-6, 8-13, 15-20, and 22-29 are in condition for allowance, and action to that end is respectfully requested. Should any issue remain that prevents allowance of the application, the Office is encouraged to contact the undersigned prior or issuance of a subsequent action.

Respectfully submitted,

Dated: 12/14/05

By: Brian Hart

Brian G. Hart
Reg. No. 44, 421
(509) 324-9256